Calculadora graficadora

W. E. S. Tubín

***Resumen*—Se documenta la realización de una calculadora graficadora, utilizando el lenguaje de bajo nivel ensamblador. El programa que ensambla es NASM. La ejecución se realizó haciendo una emulación en DOSBox.**

***Index Terms*—Consola, DOSBox, Ensamblador, NASM.**

I. Introducción

La aplicación consiste en una calculadora que permite ingresar una función polinómica de hasta grado 4, cuyas operaciones son: derivar, integrar y graficar. También resolver ecuaciones de grado no mayor a dos. Las raíces que puede encontrar son enteras signadas (positivas y negativas). Dichas funciones pueden ser ingresadas manualmente y mediante un archivo de entrada.

La implementación se realiza en ensamblador utilizando NASM version 2.13.03. Se utilizan solo instrucciones para CPU 8086 para DOS. Para probar el ejecutable se emplea DOSBox.

La interracción con el usuario se logra empleando interrupciones.

Para la graficación se emplea el modo de video 13h.

Se utiliza 16 bits para representar un número signado. Esto quiere decir que las cantidades que puede manejar están entre $-(2^{16}-1)/2$ y $+(2^{16}-1)/2$.

Los términos *función* y *polinomio* se toman como sinónimos.

II. Funcionalidades

Estas son la opciones del menú principal:

1. Derivar función: Se ingresa una función de grado máximo 4 de coeficientes enteros signados y de dos digitos como máximo. La misma se ingresa mediante una cadena e inmediatamente se calcula la *derivada* de la función ingresada. Se guarda en memoria tanto la función ingresada como la derivada calculada.
2. Integrar función: Se ingresa una función de grado máximo 4 de coeficientes enteros signados y de dos digitos como máximo. La misma se ingresa mediante una cadena e inmediatamente se calcula la *integral* de la función ingresada. Se guarda en memoria tanto la función ingresada como la integral calculada.
3. Ingresar funciones: se escribe la ruta del archivo y se deriva, integra o resuelven las funciones contenidas en él. Se guarda en memoria cada funcion así como su derivada, integral o soluciones.
4. Imprimir funciones ingresadas: Se listan las funciones en el sistema así como su derivada, integral o soluciones dependiendo de que operación se haya elejido durante su ingreso.
5. Graficar: Se grafican las funciones del sistema. Esto incluye los polinomios (las originales) y sus derivadas e integrales.
6. Resolver ecuación: Se ingresa una ecuación de grado no mayor a dos y se calcula su(s) solucion(es).
7. Reportes: Muestra un menú en el que es posible crear reportes para las funciones del sistema a las cuales se calcularon sus derivadas o integrales o bien a los que se les encontró sus raices. Para el primer caso también se muestran cinco puntos evaluados en la función.
8. Salir: Sale del programa.

Observaciones:

- El *máximo numero de funciones* posibles en el sistema es de quince. Al llegar a este número muestra un mensaje que los espacios se han llenado.
- Cada que se ingresa una funcion, vía archivo o manualmente, se realiza un *análisis léxico* de la cadena ingresada para validarla. En caso de que sea una cadena errónea se muestran los mensajes pertinentes.

III. Autómata

La validación para la cadena de entrada del polinomio, tanto manual como mediante archivo, se logra realizando un análisis lexico. Simultáneamente conforme va validado va cargando en memoria los coeficientes y exponentes si todo es correcto. Si se llega a un ";" y no ocurrio error se aumenta el contador global de funciones del sistema que puede encontrarse en el archivo `anlex2.asm` y se llama `n_pol`.

La funcion "scanner" es la que implementa el autómata realizando el análisis léxico de una función polinómica de grado 4 con coeficientes de dos digitos positivos o negativos. Un término correcto debe iniciar con "+" o "-" seguido de cero, uno o dos digitos. Luego puede venir o no una "x" o bien "x" seguido por "2" o "3" o "4". La expresión regular es la siguiente:

```
ER=(('+'|'-')(<d>?<d>)?[x^<expo>|x]?)+;
donde: <d>=0|1|2|3|4|5|6|7|8|9 y <expo>=2|3|4
```

Estos son algunos ejemplos de cadenas lexicamente correctas para este lenguaje (debe terminar en ";"):

```
+2x+1-99x  +29  +x +29;
-90x^3 +5  5x^2;
```

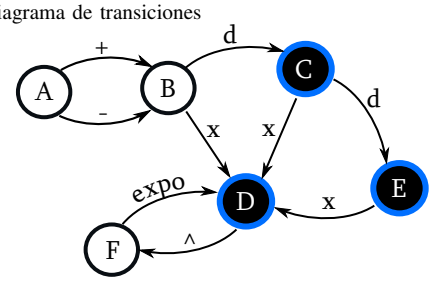
W. E. S. Tubín actualmente estudia en Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala.

Cuadro I
TABLA DE TRANSICIONES EMPLEADA

Estado/Entrada	+	-	d	x	z	expo	FDC
A	B	B	-	-	-	-	-
B	-	-	C	D	-	-	-
C	B	B	E	D	-	-	#
D	B	B	-	-	F	-	#
E	B	B	-	D	-	-	#
F	-	-	-	-	-	D	-

Las casillas con "-" son estados de error y los "#" estados de aceptación.

Figura 1. Diagrama de transiciones



No se diagramaron las transiciones desde los estados de aceptación C, D, y E hacia B, con entradas "+" y "-" por comodidad visual. De todas formas puede consultarlo en la tabla de transiciones.

IV. MODO DE VIDEO UTILIZADO

Se utilizó **Int 10/AH=00/AL=13H**. Esta es una interrupción que establece el modo de video [4]. Recibe `00h` en `AH` y el modo de video en `AL`, que en este caso es `13h`.

Este modo de video es modo grafico 40x25 de 256 colores, 320x200 pixeles y 1 página. Salidas:

- AL = bandera de modo de video
  - 20h modo >7
  - 30h modos 0-5 y 7
  - 3Fh modo 6
- AL = CRT byte del modo controlador.

V. INTERRUPCIONES

Estas son las interrupciones utilizadas.

V-A. Int 10h

V-A1. AH=02h: Es una interrupción de video y establece la posición del cursor [1]. En BH va el número de página. En DH va la fila (00h es arriba). En DL va la columna (en 00h es izquierda). Salidas:

- Nada

V-A2. AH=06h: Es una interrupción de video y desplaza la ventana hacia arriba [2]. En AL recibe número de líneas para desplazarse hacia arriba (00h = borrar toda la ventana). En BH el atributo utilizado para escribir líneas en blanco en la parte inferior de la ventana. En CH y CL la fila y columna de la esquina superior izquierda de la ventana. En DH y DL la fila y columna de la esquina inferior derecha de la ventana. Salidas:

- Nada

V-A3. AH=0Fh: Es una interrupción de video y obtiene el modo de video actual [3]. Salidas:

- AH = número de columnas de caracteres
- AL = modo de visualización
- BH = página activa .

V-B. Int 21h

V-B1. AH=4Ch: Es una interrupción de DOS. Termina con código de retorno [10]. En `AH` recibe `4ch`. En `AL` recibe el código de retorno. Salidas:

- Nada.

V-B2. AH=0Ah: Lee una cadena de caracteres desde el dispositivo de entrada estándar hasta que se presiona la tecla Enter (cuando se detecta retorno de carro). Comprueba para Ctrl-Break y Ctrl-C [11]. En DS:DX va el puntero al buffer de entrada. Este buffer debe tener un formato determinado que se muestra a continuación:

Cuadro II
FORMATO DE DOS PARA EL BUFFER DE ENTRADA [12]

Offset	Size	Descripcion
00h	BYTE	caracteres maximo que el buffer puede manejar
01h	BYTE	numero de caracteres leído, excluyendo retorno de carro
02h	n BYTES	puntero al primer caracter de la cadena leída. Esta cadena incluye el retorno de carro

El *Offset* se refiere al desplazamiento a partir del puntero inicial. Por ejemplo cuando se indica offset de 00h, en realidad se trata del primer byte del arreglo, si así lo queremos ver. Salidas:

- Buffer lleno con la entrada del usuario.

V-C. Int 1A

V-C1. AH=00h: Es una interrupción de tiempo y obtiene el tiempo del sistema [6]. Salidas:

- CX:DX =número de ciclos por segundo desde la media noche.
- AL=bandera de medianoche, distinta de cero si ha pasado la medianoche desde la última vez que se leyó.

V-D. Int 16

V-D1. AH = 01h: Es una interrupción de teclado y comprueba si hay pulsación de teclado en el buffer. [9]. Salidas:

- Establece ZF si no hay pulsación de teclado disponible
- Resetea ZF si hay pulsación de teclado disponible
- AH = Código de BIOS escaneado
- AL = Caracter ASCII

V-D2. AH = 00h: Es una interrupción de teclado y obtiene la pulsación de teclado del buffer sin eco. [8]. Salidas:

- AH = Código de BIOS escaneado
- AL = Caracter ASCII

V-E. Int 15

V-E1. AH = 86h: Es una interrupción del BIOS espera los microsegundos indicados en `CX:DX` [7]. Salidas:

- resetea CF si fue exitoso (intervalo de espera transcurrido)
- establece CF en caso de error o AH=83h espera ya en progreso
- AH = estado

VI. Archivo io.mac

Macros para entrada y salida utilizando interrupciones. Son ejecutables en DOS. Para emular DOS se utilizó DOSBox. Es ensamblado por NASM.

VI-A. TERMINAR_PROGRAMA

Salir del programa

- Entrada
  vacio nada
- Salida
  vacio nada

VI-B. PRINT

Imprimir cadena en salida estandar.

- Entrada
  dx, %1 puntero a cadena que termina en "$"
- Salida
  vacio nada

VI-C. PRINT_CHAR

Imprimir caracter en la salida estandar

- Entrada
  dl, %1 el caracter a imprimir
- Salida
  al la ultima salida de caracter

ir

VI-D. ABRIR_ARCHIVO

Abre un archivo existente.

- Entrada
  dx, %1 puntero a cadena con nombre/ruta del archivo a abrir. Este nombre debe finalizar en 0
- Salidas
  CF=0 archivo se abrio exitosamente. En AX el handle del archivo
  CF=1 ocurrio algun error. El código del error en AX

ir

VI-E. LEER_ARCHIVO

Lee archivo abierto.

- Entradas
  bx, %1 handle del archivo
  cx, %2 numero de bytes a leer
  dx, %3 buffer en donde guardar lo leido
- Salidas
  CF=0 se leyo exitosamente. En AX el numero de bytes actualmente leídos
  CF=1 ocurrio algun error. El código del error en AX

ir

VI-F. ESENAR

EScribir EN un ARchivo abierto

- Entradas
  bx, %1 handle del archivo
  cx, %2 numero de bytes a escribir
  dx, %3 puntero a array que se escribirá
- Salidas
  CF=0 se escribió exitosamente. En AX el numero de bytes actualmente escritos.
  CF=1 ocurrio algun error. El código del error en AX (05h,06h).

ir

VI-G. CERRAR_ARCHIVO

Cierra un archivo

- Entrada
  bx, %1 handle del archivo a cerrar
- Salida
  CF=0 se cerro exitosamente. AX destruido
  CF=1 ocurrio algun error al cerrar. En AX el código de error (06h).

VI-H. CREAR_ARCHIVO

Crear un archivo nuevo o truncar si ya existe.

- Entrada
  dx, %1 puntero a cadena que será nombre del archivo. Este nombre debe terminar en 0.
- Salida
  CF=0 se creo exitosamente. En AX el handle de archivo.
  CF=1 ocurrio algun error al crear En AX el código de error (03h,04h,05h).

ir

VI-I. UBICAR_CURSOR

ubicar cursor en la pantalla cuando se esta en modo texto.

- Entradas
  dh, %1 fila
  dl, %2 columna

VI-J. LIMPIAR_PANTALLA

Limpiar pantalla cuando se esta en modo texto.

- Entrada
  vacio nada
- Salida
  vacio nada

VI-K. DESPLAZAR_ARRIBA

Desplaza hacia arriba. Es como borrar la pantalla según los parámentros explicados a continuación:

- Entrada
  ch, %1 fila superior izquierda
  cl, %2 columna superior izquierda
  dh, %3 fila inferior derecha
  dl, %4 columna inferior derecha
  al, %5 número de líneas a desplazarse hacia arriba
- Salida
  vacio nada

ir

VI-L. GETCHAR

Obtiene ascii de tecla presionada sin eco, via BIOS.

- Entrada
  vacio nada
- Salida
  AL en este registro se almacena el caracter leido

VI-M. READ_FOR_BUFF_INPUT

lee cadena de entrada estandar y lo guarda un arreglo. Se refirá a este arreglo como buffInput, de aqui en adelante.

- Entradas
  dh, %1 fila
  bx, %2 columna
  dx, %3 arreglo donde guardar la entrada del usuario
- Salida
  - arreglo buffInput con la entrada del usuario. La interrupcion utilizada deja dicho arreglo en el siguiente formato, suponiedo que el usuario ingreso la cadena "cad" en la entrada estandar:

```
+--+---+---+---+---+----+---+
|41| 3 | c | a | d | \r |.. |
+--+---+---+---+---+----+---+
```

    donde:
    - [buffInput] contiene 41, indica el numero maximo de chars que puede guardar
    - [buffInput+1] contiene 3 indica el numero de chars ingresados por el usuario
    - [buffInput+2] contiene "c" es el primer char de la cadena ingresada. La siguientes dos posiciones tiene "a" y "d"
    - [buffInput+5] contiene el retorno de carro. El contenido del resto del arreglo esta indefinido

VI-N. READ_FOR_BUFF_INPUT

Lee cadena de entrada estandar y lo guarda un arreglo. NO necesita de fila y columna.

- Entradas
  dx, %1 arreglo donde guardar la entrada del usuario
- Salida
  - arreglo buffInput con la entrada del usuario.

VI-Ñ. PRINT

Imprimir cadena en salida estandar en una fila y columna especificada.

- Entrada
  dh, %1 fila
  dl, %2 columna
  dx, %3 puntero a cadena que termina en "$"
- Salida
  vacio nada

VI-O. GETCHAR

Obtiene ascii de tecla presionada, aunque se emplea para parar el programa, hasta que el usuario presione alguna tecla, mostrando un mensaje que venga al caso.

- Entrada
  %1 Puntero a cadena que se mostrara.
- Salida
  AL en este registro se almacena el caracter leido

VII. ARCHIVO ANLEX2.ASM

Contiene la funcion "scanner" que realiza el analisis lexico de una funcion polinomica de grado 4 con coeficientes de dos digitos positivos o negativos. Un termino correcto debe iniciar con "+" o "-" seguido de cero, uno o dos digitos. Luego puede venir o no una "x" o bien "x" seguido por "2" o "3" o "4". La expresion regular es la siguiente:

```
ER=(('+'|'-')(<d>?<d>)?[x^<expo>|x]?)+;
donde: <d>=0|1|2|3|4|5|6|7|8|9 y <expo>=2|3|4
```

Estos son algunos ejemplos de cadenas lexicamente correctas para este lenguaje (debe terminar en ";"):

```
+2x+1-99x  +29  +x +29;
-90x^3 +5  5x^2;
```

VII-A. scanner

Realiza un analisis lexico de la cadena terminada en ";". Muestra fila y columna si ocurre error. Muestra exito si se reconoce la cadena. Guarda los exponentes y coeficientes en una arreglo de estructuras.

- Parametro
  [BP+6] puntero al primer elemento del arreglo de chars a analizar
  [BP+4] puntero a estructura que guarda coeficientes y exponentes y otros datos de la funcion.
- Variables Locales

1. [BP-2] fila
2. [BP-4] columna
3. [BP-6] entrada
4. [BP-8] estado
5. [BP-0xA] exponente
6. [BP-0xC] primer char ascii de un coeficiente
7. [BP-0xE] segundo char ascii de un coeficiente
8. [BP-0x10] tercer char ascii de un coeficiente
9. [BP-0x12] numero de chars que forman el coeficiente
10. [BP-0x14] contador de terminos por funcion

VII-B. scanner_f

Lo mismo que scanner solo que trabaja con un archivo.

- Parametro
  - [BP+6] handle del archivo de donde lee
  - [BP+4] puntero a estructura que guarda coeficientes y exponentes y otros datos de la funcion.
- Variables Locales
  1. [BP-2] fila
  2. [BP-4] columna
  3. [BP-6] entrada
  4. [BP-8] estado
  5. [BP-0xA] exponente
  6. [BP-0xC] primer char ascii de un coeficiente
  7. [BP-0xE] segundo char ascii de un coeficiente
  8. [BP-0x10] tercer char ascii de un coeficiente
  9. [BP-0x12] numero de chars que forman el coeficiente
  10. [BP-0x14] contador de terminos por función

VII-C. rutinaDeError

Imprime un mensaje de error lexico.

- Parametros
  - [bp+8] Fila en binario
  - [bp+6] Columna en binario
  - [bp+4] El caracter ascii que causo el error
- Entrada
  - bx puntero a cadena que describe el error
- Salidas
  - vacio nada

VII-D. convNumStrParaNumBin

Convertir numero en string a numero en binario para el manejo interno como cantidad de la representacion numerica de la cadena. Se utilizó para leer la cadena directamente de la pila.

- Entradas
  - di puntero al primer elemento del arreglo de caracteres (cadena) a convertir. Cada caracter debe estar a cada dos bytes. Esto es porque se utilizó para leer de la pila cuyos elementos son de 2 bytes.
  - cl tamaño de la cadena apuntada por bx
- Salida
  - ax el numero de 1 word en binario. En complemento a dos si es negativo.
- Variable local
  - [BP-2] variable temporal (temp) que guarda la cantidad que finalmente se guarda en ax al terminar el proceso

VII-E. convNumStrParaNumBin_arr

Igual que "convNumStrParaNumBin" solo que el array de caracteres deben estar a cada byte no a cada dos, esto para que no necesariamente deba leerse desde la pila.

- Entradas
  - di puntero al primer elemento del arreglo de caracteres (cadena) a convertir.
  - cl tamaño de la cadena apuntada por di
- Salida
  - ax el numero de 1 word en binario. En complemento a dos si es negativo.
- Variable local
  - [BP-2] variable temporal (temp) que guarda la cantidad que finalmente se guarda en ax al terminar el proceso

VII-F. printNumBin

Imprimir numero binario en la salida estandar

- Entradas
  - ax el numero binario (word) a imprimir
- Salidas
  - vacio nada

VII-G. printNumBin_mas

Lo mismo que printNumBin solo que los positivos les antepone el signo positivo "+"

- Entradas
  - ax el numero binario (word) a imprimir
- Salidas
  - vacio nada

VIII. ARCHIVO PLOTER.ASM

VIII-A. dibujar_plano

dibuja los ejes X y Y del plano cartesiano

- Parámetros
  - [BP+6] posición x del origen
  - [BP+4] posición y del origen

VIII-B. putPixel

Pinta un pixel en cordenada y color dado. Las coordenadas son con respecto al origen del plano establecido durante la llamada ha "dibujar_plano".

- Entradas
  - dl color
  - bx coordenada x
  - ax coordenada y

VIII-C. *dibujar_linea_horizontal*

dibuja una linea horizontal

- Parámetros
  - [BP+10] color
  - [BP+8] ancho
  - [BP+6] x
  - [BP+4] y

VIII-D. *dibujar_linea_vertical*

Dibuja una linea vertical

- Parámetros
  - [BP+10] color
  - [BP+8] alto
  - [BP+6] x
  - [BP+4] y

VIII-E. *dibujar_rectangulo*

Dibuja un rectangulo relleno de un color dado.

- Parámetros
  - [BP+12] color
  - [BP+10] ancho
  - [BP+8] alto
  - [BP+6] x
  - [BP+4] y

IX. Archivo rep.asm

IX-A. *printPol_f*

Imprimir polinomio en archivo

- Parametros
  - [BP+6] handle del archivo
  - [BP+4] puntero al inicio del arreglo que contiene la expresion polinomica

IX-B. *printCoef_f*

Imprimir coeficiente en archivo

- Parametros
  - [PB+8] handle del archivo
  - [BP+6] coeficiente en binario (word)
  - [BP+4] exponente en binario (byte)

IX-C. *printNumBin_f*

Imprimir numero binario en archivo

- Parametro
  - [BP+6] valor para saber si se quiere imprimir los positivos anteponiendo un "+". Si tiene el valor de POS_CON_MAS se imprime el "+"
  - [BP+4] handle del archivo
- Entradas
  - ax el numero binario (word) a imprimir
- Variables locales
  - [BP-2] un temporal para ax
  - [BP-4] un temporal para bx
  - [BP-6] un temporal para cx

X. Archivo calgra.asm

X-A. *evalPol*

Evaluar polinomio de 5 terminos como maximo

- Entradas:
  - di puntero a estructura que contiene el polinomio a evaluar
- Parámetros
  - [PB+4] número a evaluar de un word (x)
- Variables locales
  - [BP-2] temporal que al final será ax
  - [BP-4] contador para bucle mayor
- Salidas
  - ax el resultado de la evaluación (y)

X-B. *printInfoEsquina*

imprimir cantidad de polinomios actuales en el sistema y cuanto es el maximo que puede ingresarse. Esto en la esquina inferior derecha.

Crear reportes.

- Variables locales
  - [BP-2] handle de archivo ya sea el creado para polinomios del sistema o bien el creado para ecuaciones resueltas. Es uno o el otro nunca ámbos.
  - [BP-4] contador del bucle
  - [BP-6] para llevar la letra de cada funcion
  - [BP-8] temporal para la evaluacion de puntos
  - [BP-10] igual que el anterior

X-C. *setOrientacionPol*

Establecer si un polinomio de grado 1,2,3 o 4, es positivo o negativo. Para uso exclusivo de rutina `conocerGrado`.

- Entradas
  - bx coeficiente del termino de exponenete mayor
- Salidas
  - ah POL_POSITIVO si positivo y POL_POSITIVO-1 si negativo

X-D. *conocerGrado*

Saber el grado de un polinomio

- Entradas
  - di puntero a estructura que contiene el polinomio
- Salidas
  - al grado del polinomio
  - ah POL_POSITIVO si positivo y POL_POSITIVO-1 si negativo

X-E. *Graficar polinomios*

- Variables locales
  - [bp-2] contador temporal para bucle mayor
  - [bp-4] limite inferior del rango a graficar (x0)
  - [bp-6] limite superior del rango a graficar (x1)
  - [bp-8] factor por el cual se dividen los puntos en el eje Y, dependiendo si se trata de un polinomio de 1,2,3 o 4 grado, esto para que se logre la visualización más adecuada de la gráfica considerando que se tiene solo 200 pixeles del modo de video para el eje Y.

X-F. *printIntegral*

Imprime una integral en salida estandar. Basicamente es lo mismo que printPol, la diferencia es que antes de hacer la llamada a printPol imprime la notacion de integrales.

- Entrada
  - si polinomio integrado

X-G. *printDerivada*

Imprime una derivada en salida estandar. Basicamente es lo mismo que printPol, la diferencia es que antes de hacer la llamada a printPol imprime la notacion de derivadas.

- Entrada
  - si polinomio derivado de otro polinomio

X-H. *printPolinomio*

Imprime una integral en salida estandar. Basicamente es lo mismo que printPol, la diferencia es que antes de hacer la llamada a printPol imprime la notacion de funcion identificandola con una letra mayuscula.

- Entrada
  - di polinomio original que no es derivada ni integral de otro polinomio

X-I. *printCoef*

Imprimir coeficiente en la salida estandar

- Parametros
  - [BP+6] coeficiente en binario (word)
  - [BP+4] exponente en binario (byte)

X-J. *printPol*

Imprimir polinomio en salida estandar

- Parametros
  - [BP+4] puntero al inicio del arreglo que contiene la expresion polinomica

X-K. *integrarPol*

Integra una polinomio de 5 terminos como maximo

- Entrada:
  - di puntero a estructura que contiene el polinomio a integrar.
  - si puntero a estructura donde guardar la integración.

X-L. *integrarPol*

Integra una polinomio de 5 terminos como maximo

- Entrada:
  - di puntero a estructura que contiene el polinomio a derivar.
  - si puntero a estructura donde guardar la derivación.

X-M. *ing*

Opcion ”ingresar” del menú principal, cuyo objetivo es cargar un archivo con expresiones polinómicas terminadas en ”;”

- Variables Locales
  - [BP-2] guarda el handle del archivo abierto.

XI. Conclusión

El lenguaje ensamblador es de bajo nivel y ayuda a compreder mejor el funcionamiento de los registros y memoria de una computadora. Así mismo DOS escribir un programa en ensamblador, para DOSBox ayuda a tener una mejor panorámica sobre las bases de sistemas operativos mas actuales como MS-Widonws.

Referencias

[1] Ralf Brown's. *VIDEO - SET CURSOR POSITION*, Disponible: `http://www.ctyme.com/intr/rb-0087.htm`

[2] Ralf Brown's. *VIDEO - SCROLL UP WINDOW*, Disponible: `http://www.ctyme.com/intr/rb-0096.htm`

[3] Ralf Brown's. *VIDEO - GET CURRENT VIDEO MODE*, Disponible: `http://www.ctyme.com/intr/rb-0096.htm`

[4] Ralf Brown's. *VIDEO - SET VIDEO MODE*, Disponible: `http://www.ctyme.com/intr/rb-0069.htm`

[5] Ralf Brown's. *TELETYPE OUTPUT*, Disponible: `http://www.ctyme.com/intr/rb-0106.htm`

[6] Ralf Brown's. *TIME - GET SYSTEM TIME*, Disponible: `http://www.ctyme.com/intr/rb-2271.htm`

[7] Ralf Brown's. *BIOS - WAIT (AT,PS)*, Disponible: `http://www.ctyme.com/intr/rb-1525.htm`

[8] Ralf Brown's. *KEYBOARD - GET KEYSTROKE*, Disponible: `http://www.ctyme.com/intr/rb-1754.htm`

[9] Ralf Brown's. *KEYBOARD - CHECK FOR KEYSTROKE*, Disponible: `http://www.ctyme.com/intr/rb-1755.htm`

[10] Ralf Brown's. *EXIT - TERMINATE WITH RETURN CODE*, Disponible: `http://www.ctyme.com/intr/rb-2974.htm`

[11] Vitaly. *Buffered Input*, Disponible: `http://vitaly_filatov.tripod.com/ng/asm/asm_010.11.html`

[12] Ralf Brown's. *BUFFERED INPUT*, Disponible: `http://www.ctyme.com/intr/rb-2563.htm#Table1344`

[13] Jones ICS. *File Processing*, Disponible: `http://fleder44.net/312/notes/16Files/Index.html`

[14] Wikipedia. *Archivos COM*, Disponible: `https://es.wikipedia.org/wiki/Netwide_Assembler`

[15] Wikipedia. *Archivos COM*, Disponible: `https://es.wikipedia.org/wiki/Archivo_COM`

W. E. S. Tubín (201213139) nació en Guatemala, Guatemala. Estudia ingeniería en Ciencias y Sistemas en la Universidad de San Carlos de Guatemala.